CIS30E Project Documentation by Michelle Umali

This project encompasses a simple program for testing a low-level DoS attack and seeing how it affects computer resources. A port scan script, a low-level Dos attack script, and 2 computer resources scripts are presented in a GUI. Program objectives include assessing and measuring program performance, applying asynchronous programming, using parallel processing or multithreading, and implementing a design pattern.

The user is able to choose among 4 category buttons on the menu from the GUI with an additional 2 submit buttons for the manual entries. Manual entry of the ip address is needed to run the port scan. A submit button must be clicked to submit the manual entry. A separate button is to start the port scan. Manual entries of the source ip address, target ip address and port number is needed for the low-level DoS attack. A submit button must be clicked to submit these 3 pieces of data while a separate button is for starting the low-level DoS scan. The third category button is to see computer resources which are the download, upload and ping speed, the total storage, used storage and free storage, and the cpu % and ram % utilization. The fourth category button is for the live plot of the cpu % and ram % utilization.

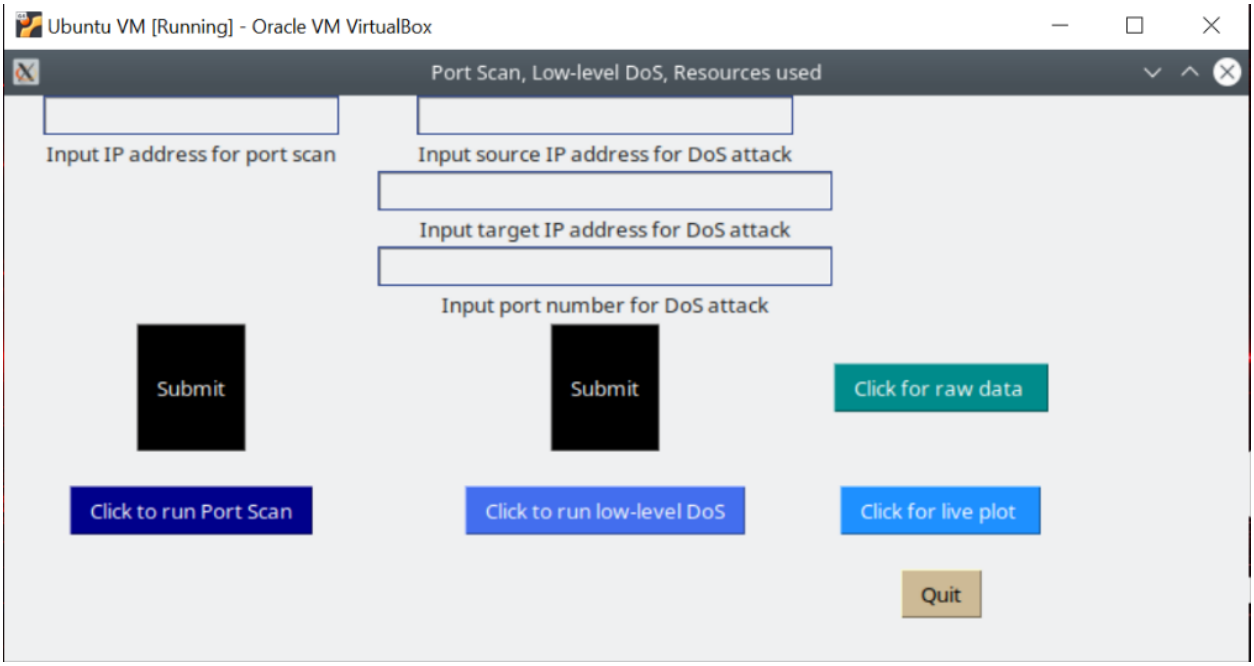
Libraries or modules include:

asyncio for asynchronous programming	multithreading for implementing multithreaded program
datetime for making a datetime object	os for script execution from button clicked
ipaddress for ip address input validation	psutil for retrieving info on system utilization
matplotlib for graphing	scapy for sending network packets
matplolib.backends - backend_tkagg.FigureCanvasTkAgg is the interface between the Figure and Tkinter canvas	shutil for memory usage statistics
matplolib.backends.backend_tkagg - NavigationToolbar2Tk is a built-in toolbar for the graph figure	socket provides the BSD socket interface
matplotlib.dates is for date plotting capabilities	speedtest for testing internet bandwidth
matplotlib.figure - Figure is the area where the graph is drawn	tkinter for the graphical user interface
matplotlib - Style is for which backend the matplotlib will use	tkinter.ttk for access to Tk themed widget set
datetime.timedelta is a duration expressing the difference between two time instances	threading for creating, controlling and managing threads
multiprocessing for multiple concurrent processing	time for measuring program execution duration

Concurrency exists in the port scan script run with multithreading in the GUI and in a separate script for port scan not presented in the GUI written with multiprocessing. The full range of ports from 1 to 65535 are scanned which necessitate multithreading or multiprocessing. The retrieval of computer resources data use asynchronous programming since it takes longer for the download and upload speeds to process versus the storage, cpu and ram data.

One limitation among others of the program is the delay it takes to calculate the bandwidth. The port scan could be run on more than one device at the same time. Also, the GUI lacks the ability to run more than one program at a time. The retrieval of the ip address from the mvc design is done by creating a text file to read it in the file. Lastly, measurements for a DoS attack are needed to conclude whether there is an attack going on.

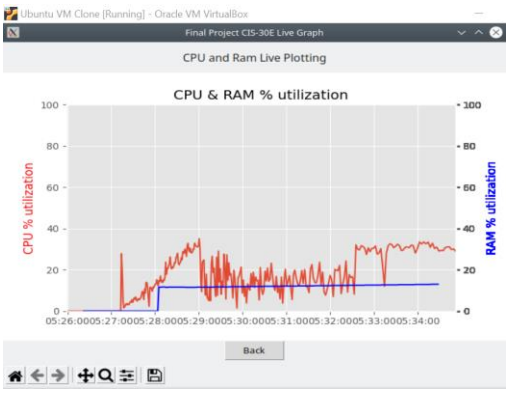
To improve the limitations would involve allowing for more than one category button to be pressed at a time. Alter the mvc design pattern to allow for more than one device port scan at a time. Add a back button to get back to the main menu. Create the proper object syntax from the mvc design pattern to retrieve the ip address without creating a text file. Strategize a script to gauge whether a DoS attack is actually occurring.

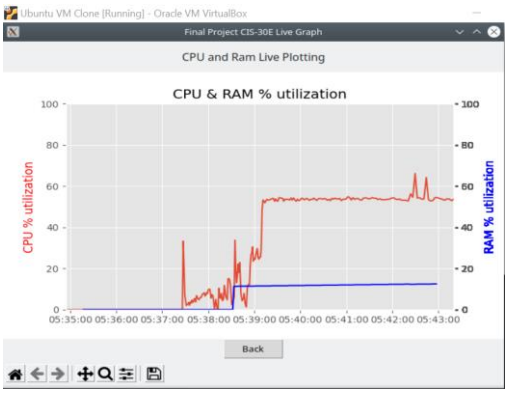


BASELINE CPU & RAM % utilization – before low-level DoS attack

Left:

Victim Ubuntu virtual machine, 8gb ram, 2 processors





Right:

CPU & RAM % utilization – during low-level DoS attack started around 5:39

Time measurement

cProfile was applied to the multiprocessing script and the time spent was on the 'built-in method posix.waitpid'(2.495 seconds).

```
        4    2.495    0.624    2.495    0.624 {built-in method posix.waitpid}
```

cProfile was applied to the multithreading script and most of the time was spent on the 'method 'acquire' of '_thread.lock' objects'(26.437 seconds) where this is the bottleneck.

```
327672   26.437    0.000   26.437    0.000 {method 'acquire' of '_thread.lock' objects}
```

PSEUDOCODE

GUI script

This program is a Model-View-Controller design pattern for Tkinter

This lays out the graphical user interface for the 3 components of the overall program

In the class controller the model and view are initialized

Event handlers are defined

Labels are defined for the ip address, source ip address, target ip address and port number entry boxes

Buttons are defined for Submit, Quit, Run port scan, run dos attack, run raw data and run live plot

Actions for when buttons are pressed are defined

Functions related to internal changes are defined with the entries stored in a list

In the View class the interface is set up for the view

One entry box and label made for the ip address to be port scanned

One button to submit the above info

Another button to run the port scan

Three entry boxes and labels made for the source and target ip address and the port number

One button made to submit

Another button to run the low-level DoS attack

Two more buttons made to see the raw resources numbers and/or the live plot

The last button set is the quit button

In the class View the frame, grid, pointer, entry text and label text are initialized

The widgets are loaded

Getters are made to return the entry text

Setters are made to set the entry text

In the class model the pointer and model are initialized

Functions for internal changes are defined

Main is defined with root, frame, the app and the mainloop

Main is called

Multiprocessing port scan script

Initialize ports list and minimum and maximum port range

Read text file to retrieve ip address to scan

Input validation if ip address is valid

Class Process is defined

A function is defined to scan all ports

Allow 0.5s for every port to scan and connect to

Append open ports to list

Print any open ports and associated services

Print is service not found for port

4 process objects are instantiated

The process is started

Stop execution once process is complete

The time execution is printed

If no open port, print no open ports

Multithreading port scan script

Threads list initialized

Global ip variable initialized

Read text file to retrieve ip address to scan

Input validation if ip address is valid

Function defined to connect to port

Time limit of 0.05 seconds to connect

Check If port is closed, pass

Check If port is open

Print Port open and associated service

If no associate service print open service not found for port

Function defined for scanning port

Initialize start time

For loop for threads to port scan all 65535 ports

Threads start

Append threads to list

Print all ports scanned

Print execution time

Call main

Function scan ports called

Low-level DoS attack

After pressing button on GUI, admin password required to run script

Initialize threads

Read file to retrieve source ip address, target ip address, port number

Input validation for entries

Attack function defined

Source and ip address used

Source and destination port used

Using Scapy, packet is sent

Packet number printed

Multithreaded function is defined

Jobs list is initialized

Thread object is created

Threads are appended to list

Jobs start

Jobs stop

Main called

Raw resources script

Variable for speedtest initialized

Download speed in async function

Upload speed in async function

Ping speed in async function

Storage in async function

Ram and cpu in async function

In main have each function become a task

Each task to await

Call main() in asyncio

Resources from 3 tasks are printed in a loop

Live Plot

Font initialized

Style set

Class app defined

Make constructor

Initialize parameters

Set title

Define container characteristics

Make frame for start page

Make frame for live plot page

Allow for switching frames

Make class Start Page

Make label for button

Make button for live plot

Make Graph page

Use number of points in window as parameter

Make label for plot

Make matplotlib canvas

Format x-axis to show time

Initialize x, y and second y data

Create and label plot

Set x-axis view limits

Set y-axis view limits

Create tkinter canvas using instance of FigureCanvasTkAgg

Create toolbar on Tkinter window using NavigationToolbar2Tk class

Make button that returns to previous page

Place toolbar in frame

Call animate function

Define animate function

Add new x data to list

Add new y data to list

Add new second y data to list

Remove oldest data from each list

Update plot data

Set label for y-axis

Add a secondary axis

Add a label to the secondary axis

Set a title to the graph

Allow the x-axis labels to change with the changing time

Redraw the plot

Repeat animate function after 1 second

Call app

Set size of canvas

Run mainloop to execute app

References:

Mvc tkinter design pattern

https://makeapppie.com/2014/05/23/from-apple-to-raspberry-pi-a-mvc-template-for-tkinter/

Port Scan

https://www.youtube.com/watch?v=x4AE5yOF9pE

https://initialcommit.com/blog/portscan-programming-security

Low-level DoS attack

https://www.tutorialspoint.com/python_penetration_testing/python_penetration_testing_dos_and_ddos_attack.htm

https://www.neuralnine.com/code-a-ddos-script-in-python/

Raw data

https://www.geeksforgeeks.org/test-internet-speed-using-python/

https://stackoverflow.com/questions/48929553/get-hard-disk-size-in-python

https://www.geeksforgeeks.org/how-to-get-current-cpu-and-ram-usage-in-python/

Live plot

https://stackoverflow.com/questions/63152444/how-do-i-create-a-live-graph-with-cpu-data-in-tkinter

Lanaro, Dr. Gabriele; Nguyen, Quan; Kasampalis, Sakis. Advanced Python Programming: Build high performance, concurrent, and multi-threaded apps with Python using proven design patterns (p. 168-171). Packt Publishing. Kindle Edition.